Modelica Overview

by Martin Otter

Abstract:
This slide set gives an overview about the Modelica language, including users view, libraries and a sketch of the language elements.

License

Revisions:

2009-07-17	Martin Otter (DLR-RM and Chairman of Modelica Association): First version, based on material from courses given at Technical University of Munich.
2013-08-28	Dietmar Winkler(Telemark University College) Updated information on MSL and MA

Contents

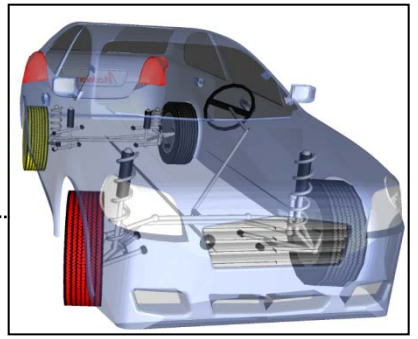
1. Modelica Introduction

Goal of **Modelica**:

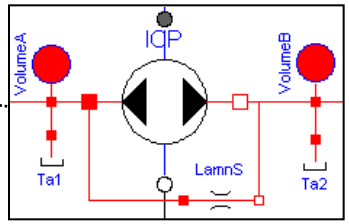
- Modeling the **dynamic behavior** of **technical systems** consisting of components from, e.g., mechanical, electrical, thermal, hydraulic, pneumatic, fluid, control and other domains in a **convenient way**.
- Models are described by **differential, algebraic**, and **discrete equations**.
- No description by partial differential equations, i.e.,
  no FEM (finite element method) and
  no CFD (computational fluid dynamics),
  but using results of, e.g., FEM programs.
- Modelica is used in industry since year 2000.

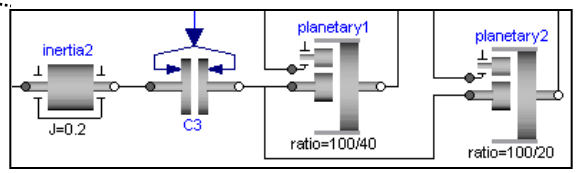
Example: **detailed vehicle model**

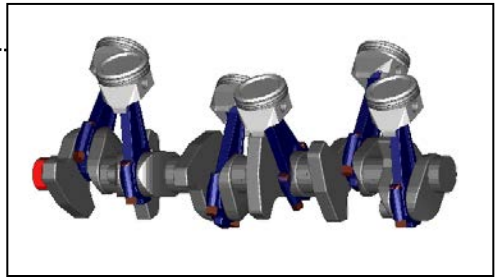
- **Vehicle dynamics** (3-dim. mechanics)
- **Drive trains** (1-dim. mechanics)
- **Hydraulics**
- **Combustion**
- **Air Conditioning**
  (Thermofluid systems)
- **Electrical/electronic systems**
- **Electrical machines**
- **Hierarchical state machines**
- **Control** (Input/output blocks, ...)

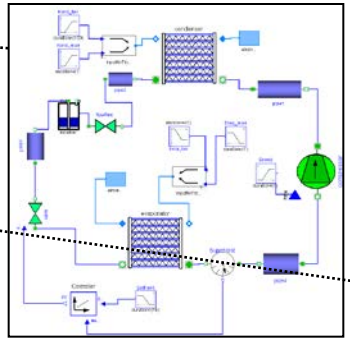


courtesy: Modelon AB

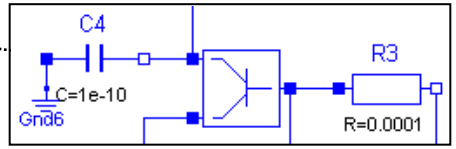


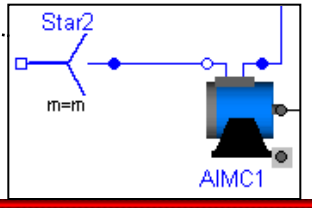


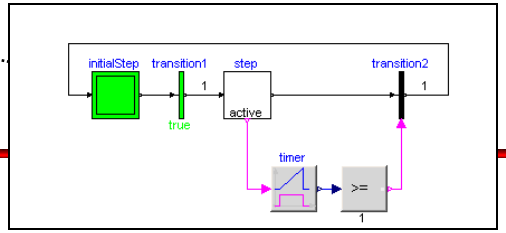




courtesy Modelon AB

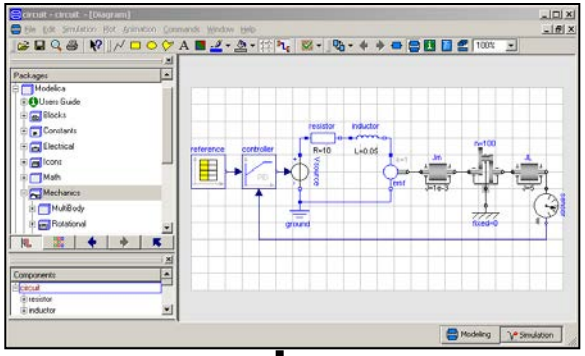






Modelica Language und Simulation-Environments

Graphical editor
for Modelica models



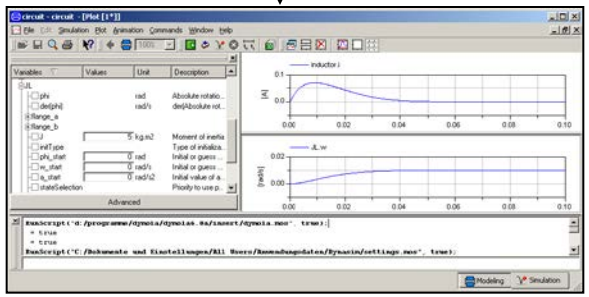
Modelica simulation environment
(free or commercial)

Textual description
on file (equations, "schematic", animation)

```
model circuit
  a;
  Modelica.Electrical.Analog.Basic.Resistor resistor(R=10) a;
  Modelica.Electrical.Analog.Basic.Inductor inductor(L=0.05) a;
  Modelica.Electrical.Analog.Basic.EMF emf(k=1) a;
  Modelica.Electrical.Analog.Sources.SignalVoltage Vsource a;
  Modelica.Electrical.Analog.Basic.Ground ground a;
  Modelica.Blocks.Continuous.LimPID controller a;
  Modelica.Mechanics.Rotational.Inertia Jm(J=1e-3) a;
  Modelica.Mechanics.Rotational.IdealGear n(ratio=100) a;
  Modelica.Mechanics.Rotational.Fixed fixed a;
  Modelica.Mechanics.Rotational.Inertia JL(J=5) a;
  Modelica.Mechanics.Rotational.Sensors.SpeedSensor sensor a;
  Modelica.Blocks.Sources.CombiTimeTable reference a;
```

Free Modelica language

Translation of Modelica models in C-Code, **Simulation**, and interactive **scripting** (plot, freq. resp., ...)



Modelica Simulation-environment
(free or commercial)

Commercial Modelica Simulation Environments (alphabetical list)

- **CATIA Systems** from Dassault Systèmes
  (based on Dymola kernel with PLM integration)
- **CyModelica** from CyDesign
- **Dymola** from Dynasim AB, Sweden
  (Dynasim was acquired by Dassault Systèmes in 2006).
- **LMS Imagine.Lab AMESim** from LMS International
- **MapleSim** from MapleSoft, Canada.
- **MathModelica** from Wolfram Research, Sweden.
- **SimulationX** from ITI GmbH, Dresden, Germany.

Free Modelica Simulation Environments (alphabetical list)

- **JModelica.org** from Lund University and Modelon AB, Sweden
  (under development; subset of Modelica is available).
- **OpenModelica** from Linköping University, Sweden
  (under development; subset of Modelica is available)

An up-to-date list of Modelica tools is available from www.modelica.org/tools

Modelica Association

- **Modelica** is a **free language** and is developed by the (non-profit) Modelica Association since 1996:

  2000: First applications

  ...

  2005: **Modelica 2.2**

  2007: **Modelica 3.0**

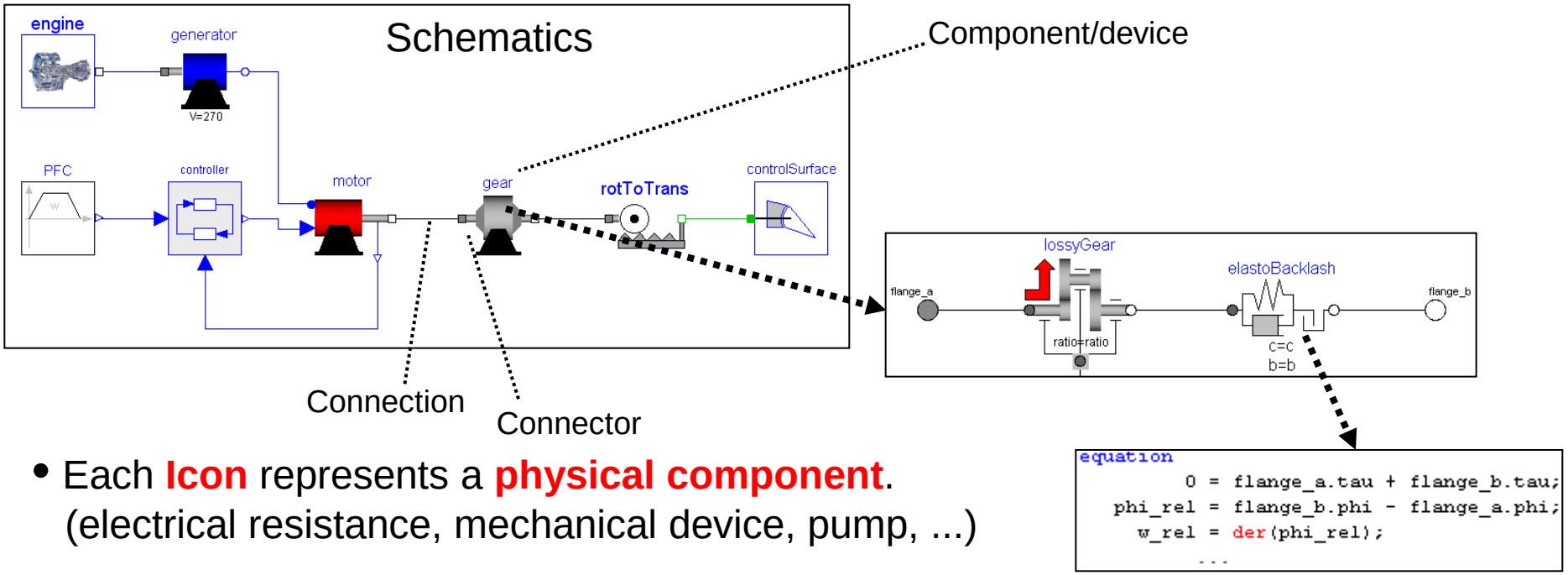
  ...

  2012: **Modelica 3.3** (current)

66th Design Meeting in Hamburg, March 2010
(after release of Modelica 3.2)

- Develops also the largest, free library for multi-domain models (**Modelica Standard Library**)
- 112 "individual" and 16 "organizational members" (interested in "active" individual members; Therefore requirement: participation at 2 Modelica Design Meetings in the last 12 months).
- 9 International Modelica Conferences (Modelica'2012 with 400 participants)
- All infos under **http://www.modelica.org** (Specification, simulation environments, free libraries, 400 papers, ...)

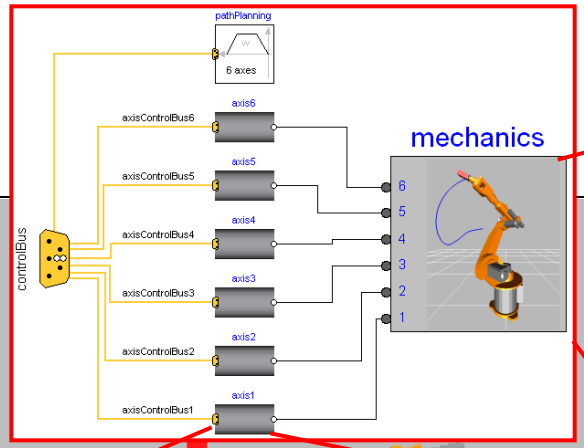
2. Modelica Users View

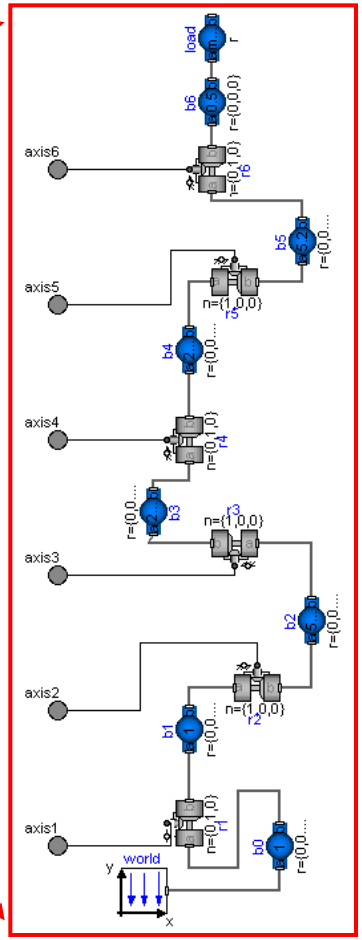


- Each **Icon** represents a **physical component**. (electrical resistance, mechanical device, pump, ...)
- A **connection line** represents the actual physical **coupling** (wire, fluid flow, heat flow, ...)
- A component consists of **connected** sub-components (= hierarchical structure) and/or is described by **equations**.
- By **symbolic** algorithms, the high level Modelica description $0 = \mathbf{f}(\dot{\mathbf{x}}(t), \mathbf{x}(t), \mathbf{y}(t), t)$ is transformed into a set of explicit differential equations: $\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), t)$

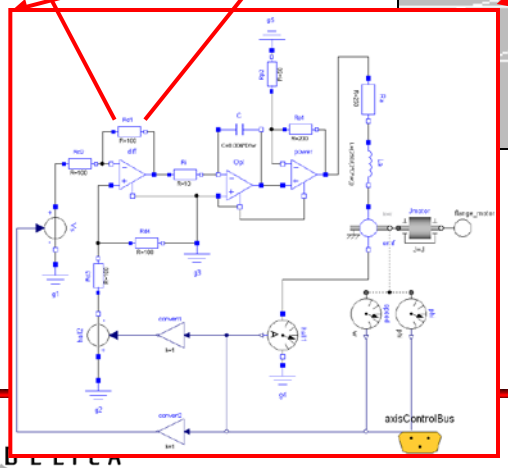
$$\mathbf{y}(t) = \mathbf{f}(\mathbf{x}(t), t)$$

Example: Industrial Robots (from Modelica.Mechanics.MultiBody.Examples.Systems.RobotR3.fullRobot)






```
model Resistor
   extends OnePort;
   parameter Real R;
equation
   v = R*i;
end Resistor;
```



1000 non-trivial **algebraic equations, 80 states.**
Faster as real-time on slow PC.

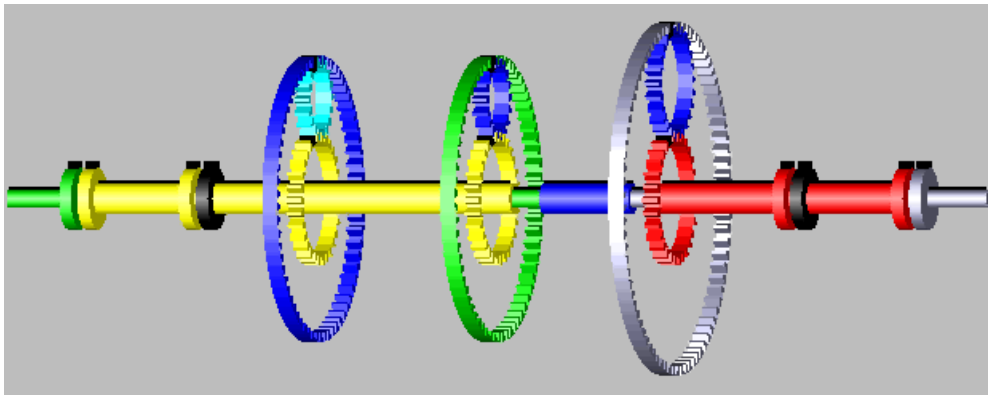
Example: Hardware-in-the-Loop Simulation of automatic gear boxes

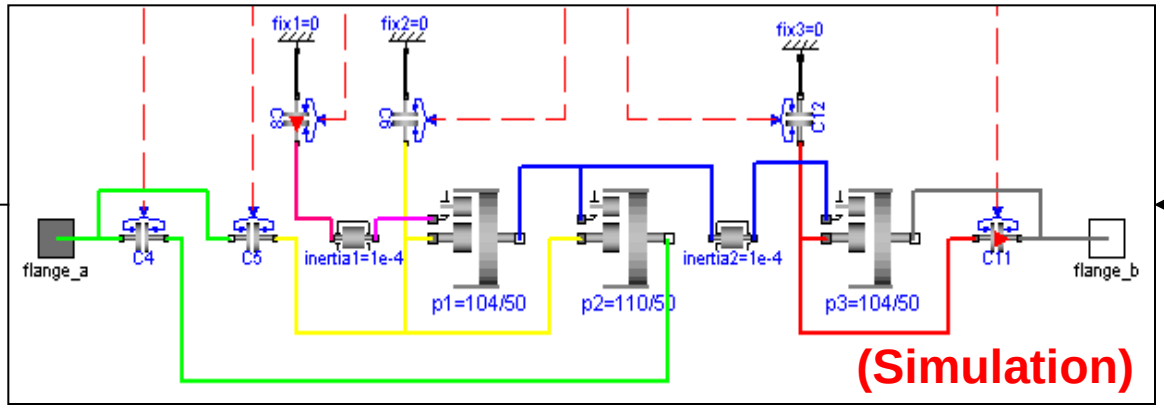
(different vehicle manufacturers)

desired pressure

Electronic Control Unit

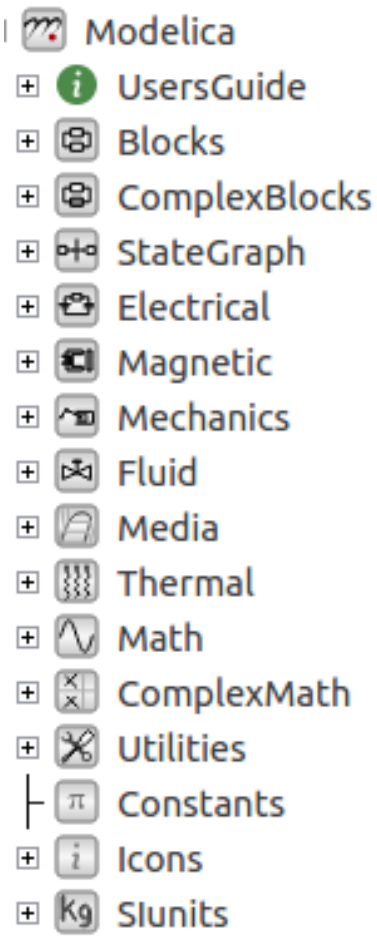
(Hardware)





+ driver + engine
+ 1D vehicle dynamics

3. Modelica Libraries



Library „**Modelica**“ is the

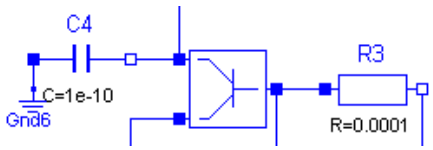
Modelica Standard Library

which is developed from the Modelica Association.
It is freely available in source code and
can be modified and used in commercial programs.

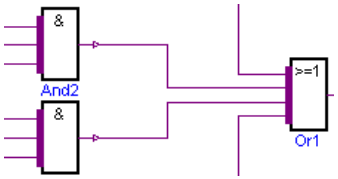
Continuous development since 1998.
Newest version 3.2.1 from August 2013:

1340 generic models
1000 functions
1450 packages (mostly media definitions)

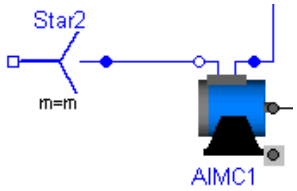
Library Modelica: Electrical and Thermal Libraries



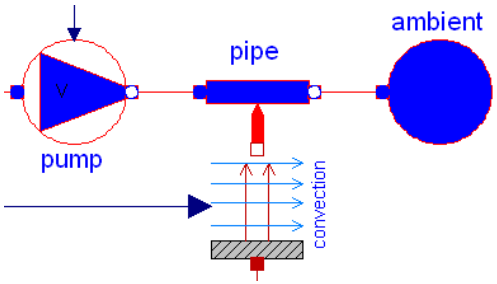
Analog electric and electronic components, such as resistor, capacitor, transformers, diodes, transistors, transmission lines, switches, sources, sensors.



Digital electrical components based on the VHDL standard, like basic logic blocks with 9-valued logic, delays, gates, sources, converters between 2-, 3-, 4-, and 9-valued logic.



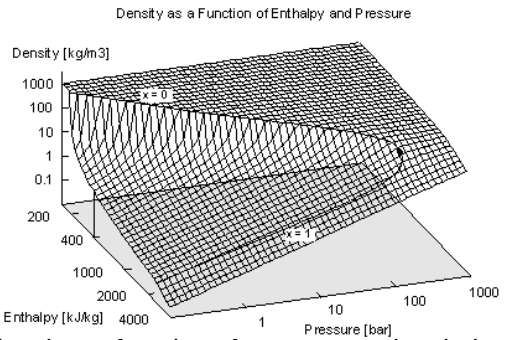
Electrical machines
(uncontrolled asynchronous-, synchronous-, DC-machines)



Simple thermo-fluid pipe flow, especially to model cooling of machines with air or water (pipes, pumps, valves, ambient, sensors, sources) and

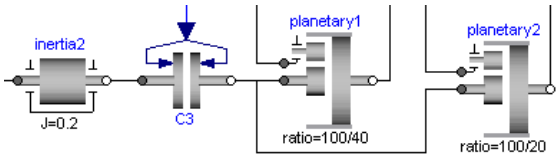
lumped heat transfer with heat capacitors, thermal conductors, convection, body radiation, sources and sensors.

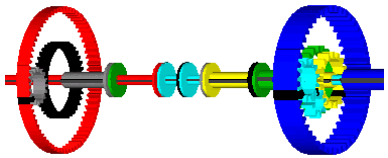
Library Modelica: Media and Mechanical Libraries



density as function of pressure and enthalpy for water

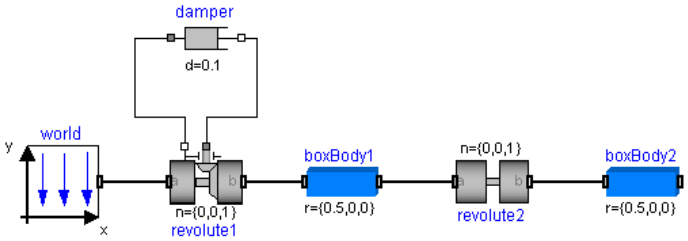
Large media library with
- 1240 gases and mixtures between these gases.
- table based media (h = h(T), etc.)
- high precision model for water (IF97)
- moist air.

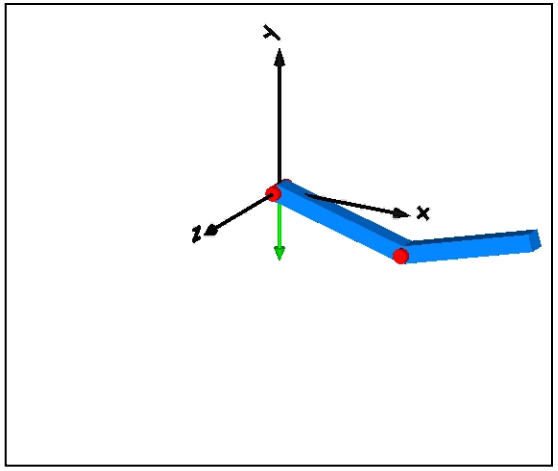


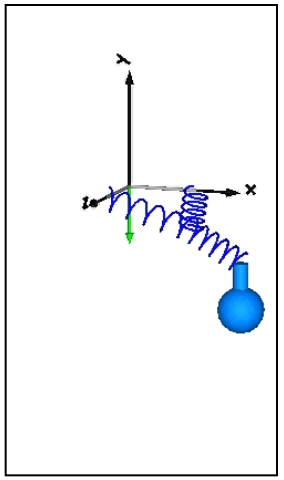


1-dim. mechanical systems, e.g., drive trains, planetary gears, convenient definition of speed/torque dependent friction (clutches, brakes, bearings, ..)

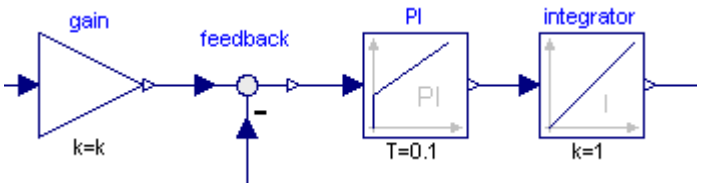
3-dim. mechanical systems consisting of joints, bodies, force and sensor elements. Joints can be driven by drive trains defined by 1-dim. mechanical system library.





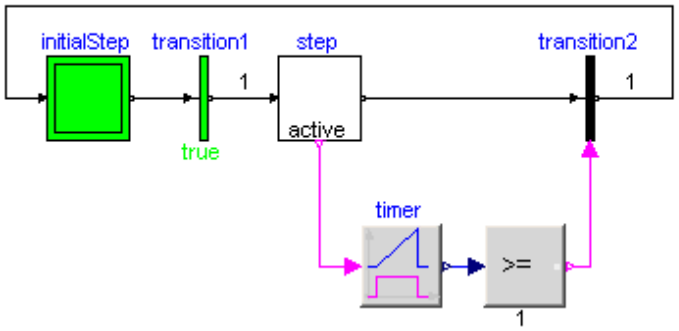


Library Modelica: Control and Script Libraries



Continuous and **discrete input/output blocks**, e.g., PI, PID, transfer function, state space, filter, logical, non-linear, routing, table source blocks

---



Hierarchical state machines with same modeling power as Statecharts. Modelica is used as synchronous action language, i.e. deterministic behavior is guaranteed (not the case for Statecharts)

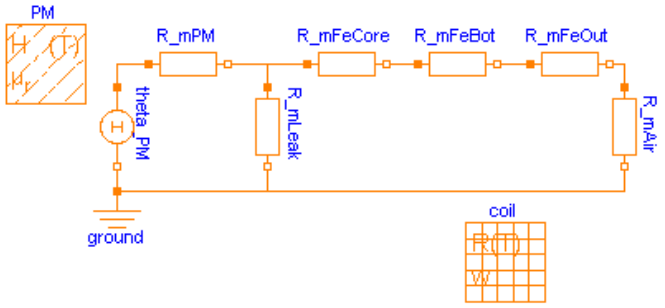
Logical blocks such as "and, or, edge, timer, ", ...

```
A = [1,2,3;
     3,4,5;
     2,1,4];
b = {10,22,12};
x = Matrices.solve(A,b);
Matrices.eigenValues(A);
```

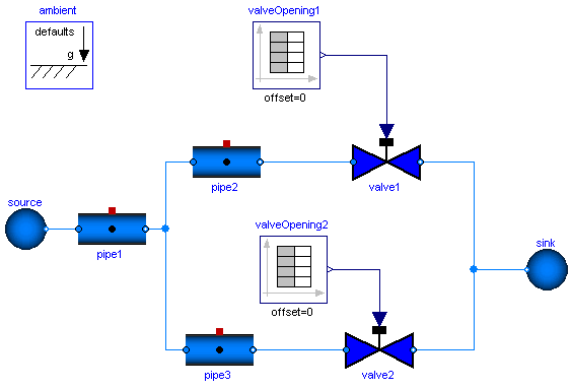
Functions on **matrices**, such as for solving linear systems, eigen and singular values etc.,

and **functions** operating on strings, streams, files, e.g., to copy and remove a file or sort a vector of strings.

Library Modelica: Sublibraries that were added in 3.1



Electro-magnetic devices with lumped magnetic networks. E.g. flux tubes, magnetic sources and sensors, magnetic materials.



General library for **fluid pipe flow** for all media of Modelica.Media

- **one** and **multiple substances**
- **one** and **multiple** (homogenous) **phases**
- **incompressible** and **compressible**

More free libraries under www.Modelica.org/libraries

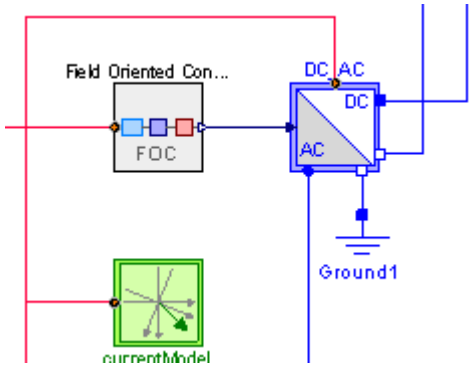
Standard conform libraries developed by the MA

Name	Description	Last Release	Last Active
ModelicaStandardLibrary	Free (standard conform) library from the Modelica Association to model mechanical (1D/3D), electrical (analog, digital, machines), thermal, fluid, control systems and hierarchical state machines. Also numerical functions and functions for strings, files and streams are included.	v3.2.1+build.2 (6 days ago)	6 days ago
Modelica_DeviceDrivers	Free (standard conform) library for interfacing hardware drivers to Modelica models. There is support for joysticks, keyboards, UDP, shared memory, AD/DA converters and other devices.	v1.1build2 (3 months ago)	23 days ago
Modelica_Synchronous	Free (standard conform) library to precisely define and synchronize sampled data systems with different sampling rates. It provides convenient to use blocks to utilize the new synchronous language elements introduced in Modelica 3.3.	v0.91 (11 months ago)	3 months ago

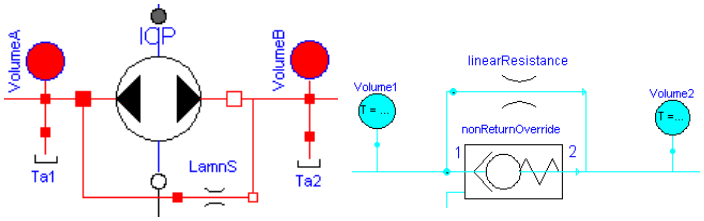
Other libraries developed by the MA

Name	Description	Last Release	Last Active
ExternalMedia	The ExternalMedia library provides a framework for interfacing external codes computing fluid properties to Modelica.Media-compatible component models.	N/A	a month ago
Modelica_EnergyStorages	Free library that contains models with different complexity for simulating of electric energy storages like batteries (single cells as well as stacks) interacting with loads, battery management systems, loads and charging devices.	N/A	5 months ago
Modelica_LinearSystems2	Free library providing different representations of linear, time invariant differential and difference equation systems, as well as typical operations on these system descriptions.	v2.3 (a year ago)	5 months ago
Modelica_StateGraph2	Free library providing components to model discrete event, reactive and hybrid systems in a convenient way with deterministic hierarchical state diagrams. Modelica_StateGraph2 is not fully Modelica compliant and will never be, since a better solution is now available with Modelica 3.3	v2.0.1 (3 years ago)	5 months ago
PowerSystems	The library is intended to model electrical power systems at different levels of detail both in transient and steady-state mode.	v0.2 (4 months ago)	23 days ago

Quickly growing number of commercial libraries. Small selection:

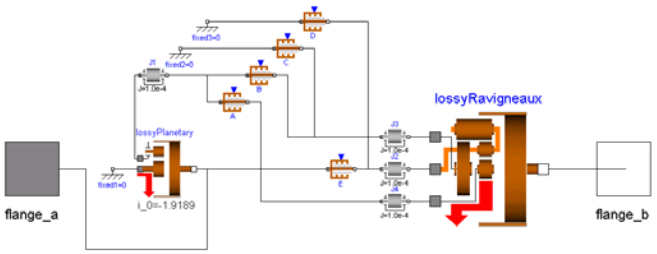


SmartElectricDrives (ATI, Austria)
Controlled electrical machines with quasi-stationary and transient models, e.g.,
controllers (voltage/frequency, field-oriented, speed/position),
power electronics (AD/DC, DC/AC, DC/DC converters, PWM),
energy storages (batteries, supercaps, fuel cells), ...



Hydraulic/Pneumatic Libraries (Modelon AB, Sweden)
Libraries to model pipe networks for oil and air. Contain all important standard components like pumps, valves, volumes, lines, sensors

---



PowerTrain (DLR-RM, Germany)
Library to model vehicle power trains and all type of planetary gearboxes. E.g. standard and planetary gears with losses, clutches with friction, flexible driveline models, automatic gearboxes,
optional 3D effects (mounting on vehicle)

4. Modelica Language Elements

Example: Definition of Capacitor

```
connector Pin
  Voltage      v;   // identical at connection
  flow Current i;   // sums to zero at connection
end Pin;

partial model TwoPin
  Pin p, n;   Voltage v;
equation
  v = p.v - n.v;
  0 = p.i + n.i;
end TwoPin;

model Capacitor
  extends TwoPin;
  parameter Capacitance C;
equation
  C*der(v) = p.i;
end Capacitor;
```

$$\frac{dv}{dt}$$

Example: Hierarchical Modelica Model

textual representation

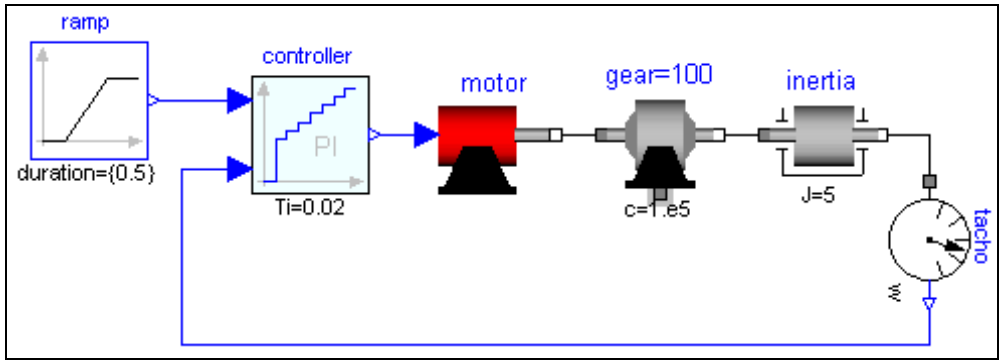
```
model MotorDrive
  PI          controller;
  Ramp        ramp;
  Motor       motor;
  Gearbox     gear(ratio = 100);
  Inertia     inertia(J = 10);
  SpeedSensor tacho;
equation
  connect(controller.y    , motor.i_ref);
  connect(motor.flange    , gearbox.flange_a);
  connect(gearbox.flange_b, inertia.flange_a);
  connect(inertia.flange_b, tacho.flange);
  connect(tacho.w         , controller.u_m);
  connect(ramp.y          , controller.u_r);
end MotorDrive;
```

Class name

Modifier

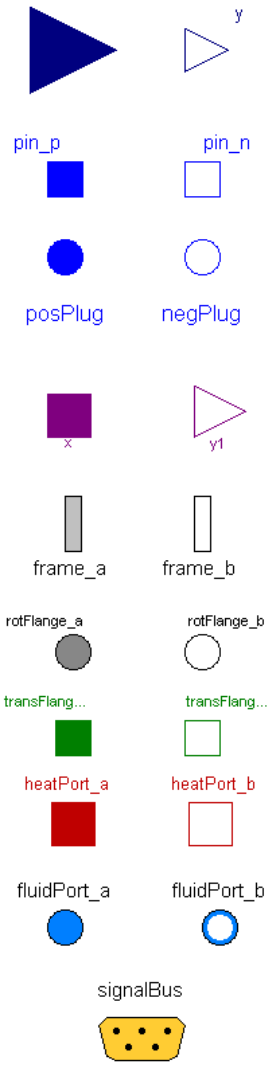
Instance name

Connector



graphical representation

Many pre-defined connectors, e.g.:



input/output signals

electrical pins

electrical plugs (multi-phase networks)

VHDL digital input/output signals

3-dim. mechanical frame

1-dim. rotational mechanical flange

1-dim. translational mechanical flange

1-dim. heat transfer

fluid port (for all media from Modelica.Media)

signal bus

Different types of variables in a connector definition

Category	*Example*	*Explanation*
input/output variable	**input** Real u	Connected variables are identical; block diagram connection restrictions
potential variable	Real v;	Connected variables are identical
flow variable	**flow** Real i;	Sum of the connected variables is zero
stream variable	**stream** Real h;	Describes bi-directional flow of matter (more complicated definition)
overdetermined variable set		Redundant set of variables with constraint equations, e.g., orientation matrix, dq0 transformation (more complicated def.)
signal bus	**explandable connector** Bus **end** Bus;	Content defined by signals connected to the connector.

Other Language Elements

- Mathematical notation for **matrices** and **arrays**
- **Arrays** not only from numbers but also **from models**
  (e.g. arrays of resistors).
- **Replaceable submodels**, e.g., to change quickly between different versions of a transmission in a vehicle system model.
- Language elements to define conveniently **discontinuous** and **variable structure** systems, e.g., to model friction or ideal switches.
- Mathematical **functions** with varying number of input/output arguments. The procedural part of Modelica is used as scripting language.
- Convenient calling of **C, Fortran**, and **Java** functions within Modelica.
- **Powerful library concept**
  (Modelica tool has enough information to find model in the file system automatically, version handling, transformations between versions, ...).